



Week 11

Computational Intelligence: Genetic Algoritihm

Mudrik Alaydrus
Faculty of Computer Sciences
University of Mercu Buana, Jakarta

mudrikalaydrus@yahoo.com



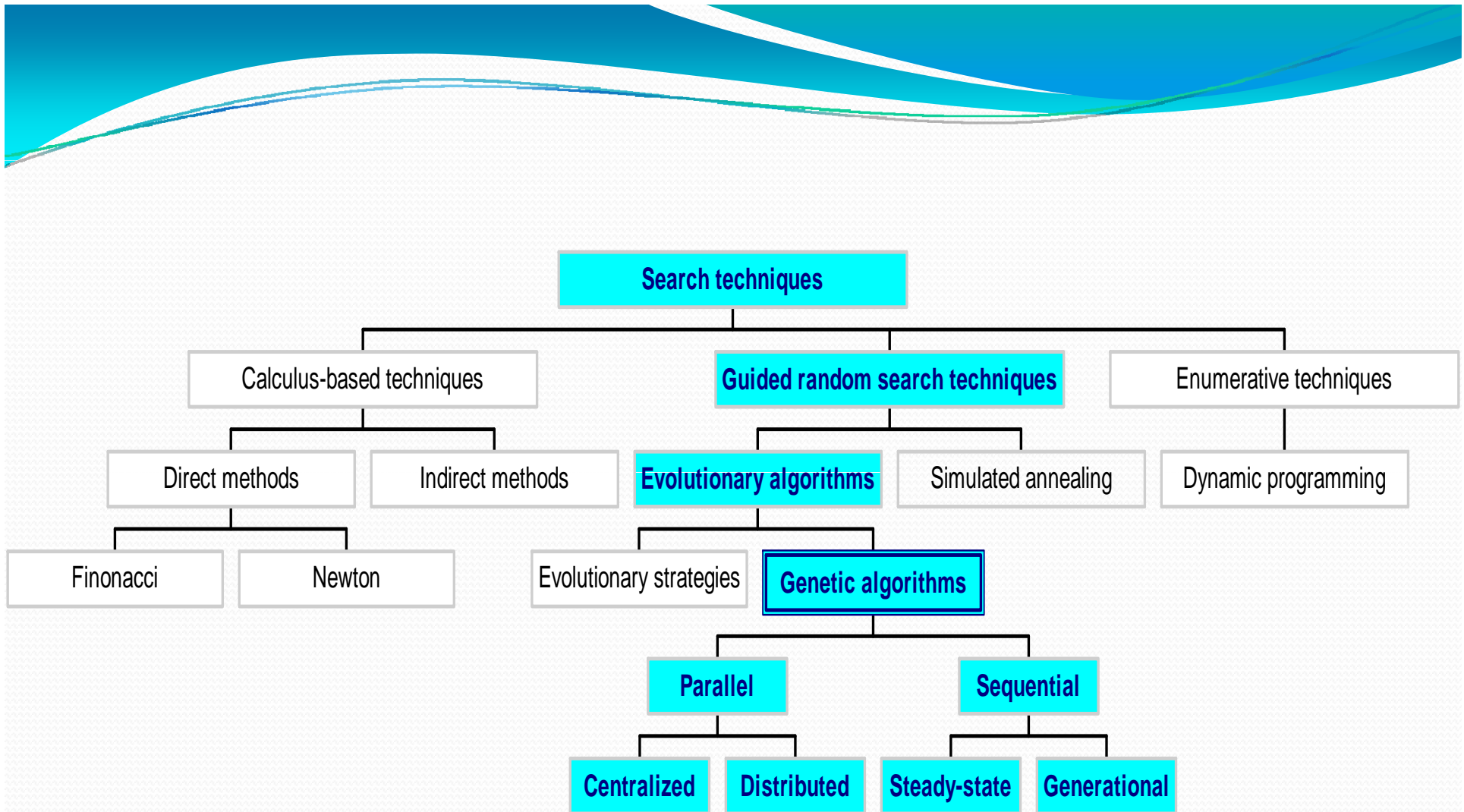
Outline


Origin, biological background

Basic Procedures

Implementation

Examples



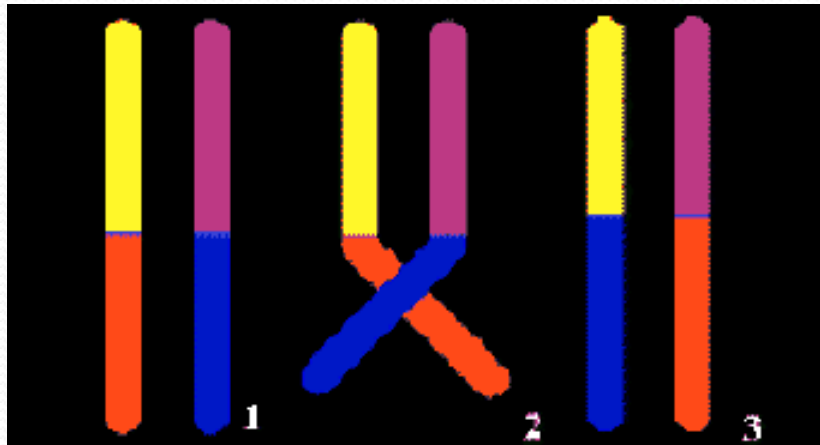
- 
- Directed search algorithms based on biological evolution
 - Developed: USA in the 1970's
 - Pioneers: J. Holland, K. DeJong, D. Goldberg
 - Typically applied to:
 - discrete optimization
 - Attributed features:
 - not too fast
 - good heuristic for combinatorial problems
 - Special Features:
 - Traditionally emphasizes combining information from good parents (crossover)
 - many variants, e.g., reproduction models, operators

Biological Background – Chromosomes

- Genetic information is stored in the chromosomes
- Each chromosome is build of DNA
- Chromosomes in humans form pairs
- There are 23 pairs
- The chromosome is divided in parts: genes
- Genes code for properties
- The possibilities of the genes for one property is called: allele
- Every gene has an unique position on the chromosome: locus

Biological Background – Reproduction

- During reproduction “errors” occur
- Due to these “errors” genetic variation exists
- Most important “errors” are:
 - **Recombination (cross-over)**
 - **Mutation**



Biological Background – Natural selection

- The origin of species: “Preservation of favourable variations and rejection of unfavourable variations.”
- There are more individuals born than can survive, so there is a continuous struggle for life.
- Individuals with an advantage have a greater chance for survive: survival of the fittest.
- **Mathematical expresses as fitness: success in life**



Genetics and evolution → adaptation of population to succeed
in its environment

→ the population is optimized for its environment

Gens, and its values (the so-called alleles), are combined together
to build a chromosome.

The best chromosome will survive.

There are chromosomes in a population,
we look after the best chromosome in this population.

So, chromosomes in a population are possible solution

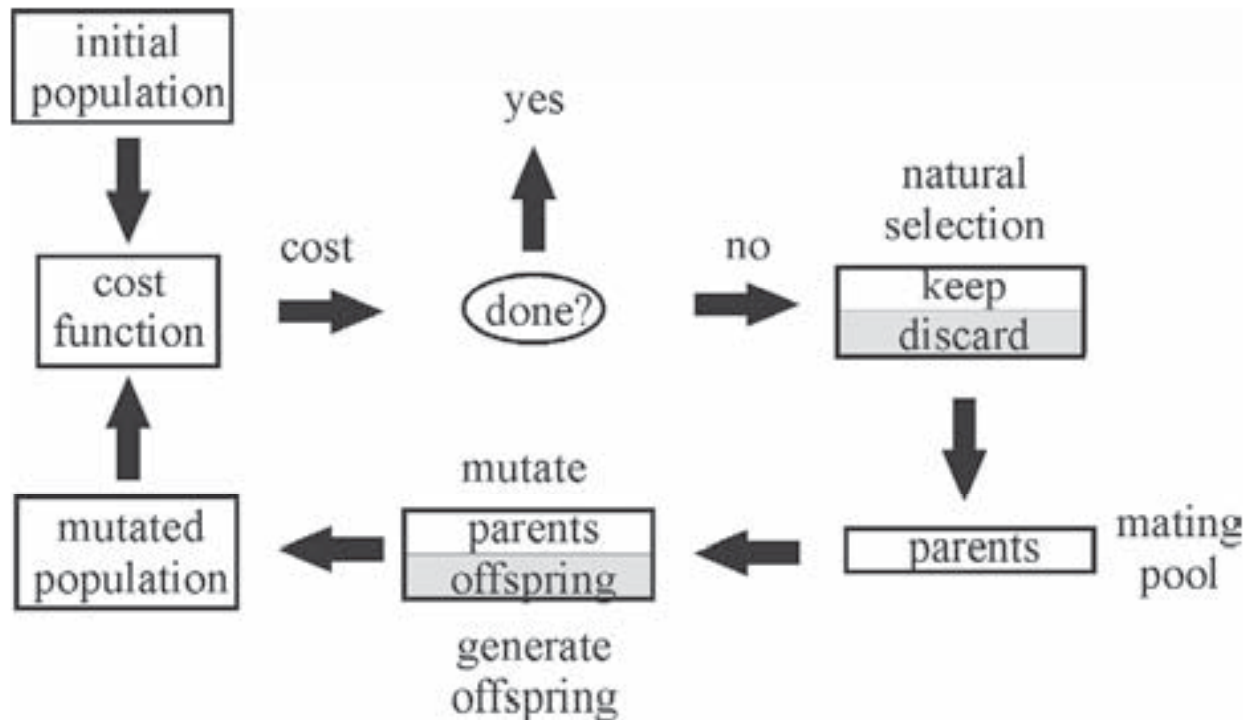
→ inputs of cost function

A basic, “no thrills” GA is quite simple and powerful.

The algorithm has the following steps:

1. Create an initial population. (observe a population)
2. Evaluate the fitness (cost function) of each population member.
(check the ‘healthiest’ individuums/chromosomes)
3. Invoke natural selection.
(choose the best chromosomes as parents)
4. Select population members for mating.
5. Generate offspring.
(cross over)
6. Mutate selected members of the population.
7. Terminate run or go to step 2.

The GA flow chart



Initial Population

a group of chromosomes

initial
population

Chromosomes could be:

- Bit strings (0101 ... 1100)
- Real numbers (43.2 -33.1 ... 0.0 89.2)
- Permutations of element (E11 E3 E7 ... E1 E15)
- Lists of rules (R1 R2 R3 ... R22 R23)
- Program elements (genetic programming)
- ... any data structure ...

The initial population is generated randomly
we must define how many chromosomes are considered (npop)
and how many variables (genes) are taken into account (nvar)

In Matlab `pop=rand(npop,nvar)`

example for npop = 8 and nvar=4 (function with 4 variables)

pop =
$$\begin{bmatrix} 0.3238 & 0.7566 & 0.9492 & 0.4134 \\ 0.9938 & 0.8282 & 0.0240 & 0.3138 \\ 0.7411 & 0.4610 & 0.2963 & 0.4225 \\ 0.1501 & 0.6775 & 0.2802 & 0.3390 \\ 0.1324 & 0.5666 & 0.9714 & 0.7557 \\ 0.1914 & 0.3403 & 0.1296 & 0.0562 \\ 0.1496 & 0.1734 & 0.4603 & 0.4604 \\ 0.5405 & 0.9526 & 0.7820 & 0.2431 \end{bmatrix}$$

Find the maximum of $f(x) = e^{-2x} \sin(3x)$

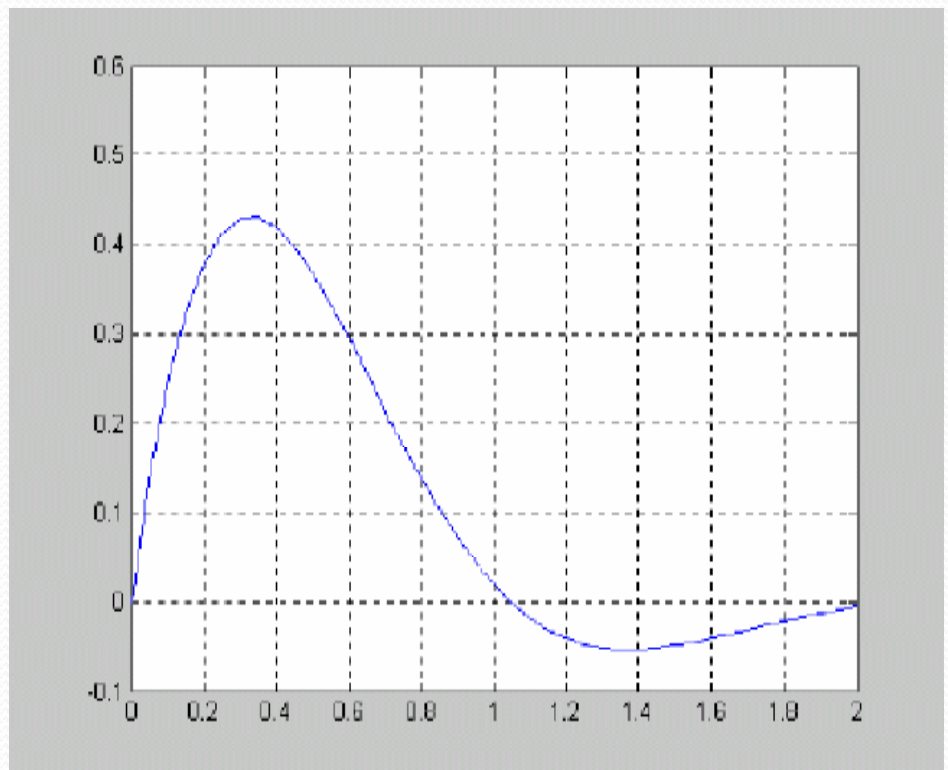
We will restrict the interval under consideration $0 < x < 1$

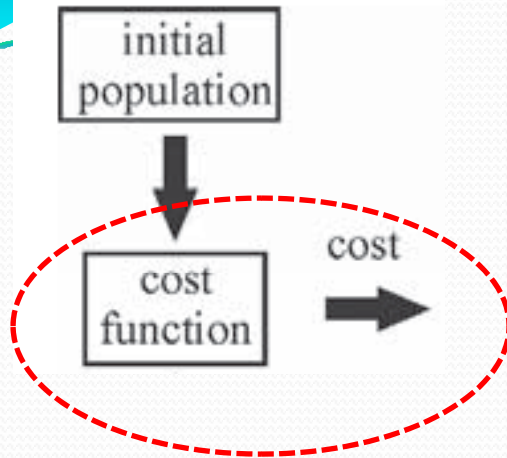
And using binary genetic algorithm with 8 bit ($= 2^8 = 256$)

→ it means 00000000 equal 0

11111111 equal 1.0

1	1	0	0	1	1	0	0	204/256 = 0.79688
0	0	0	1	1	0	0	1	25/256 = 0.097656
0	0	0	0	1	1	1	1	15/256 = 0.058594
1	1	1	0	1	0	0	1	233/256 = 0.91016
0	0	1	0	1	0	0	1	41/256 = 0.16016
1	0	1	0	0	0	1	1	163/256 = 0.63672
1	1	0	1	0	1	0	1	213/256 = 0.83203
1	0	0	0	1	0	1	0	138/256 = 0.53906
0	0	1	0	1	1	0	1	45/256 = 0.17578
1	1	0	0	0	0	1	1	195/256 = 0.76172





Formulation the cost function → extremely important step

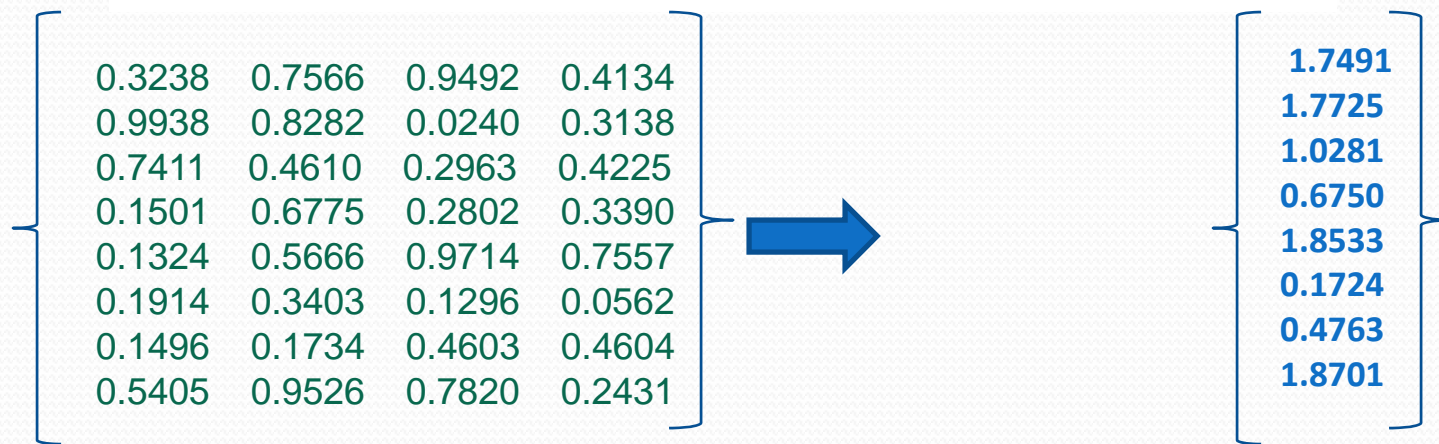
→ the cost function will be called many times

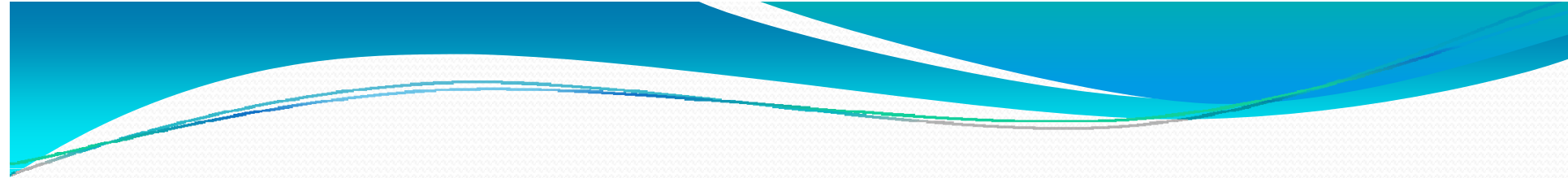
so that trade-off : accuracy vs evaluation time

Cost function : single-objective or multi-objective optimization

An example of a cost function is

$$cost = f(x_1, \dots, x_N) = \sum_{n=1}^N x_n^2$$



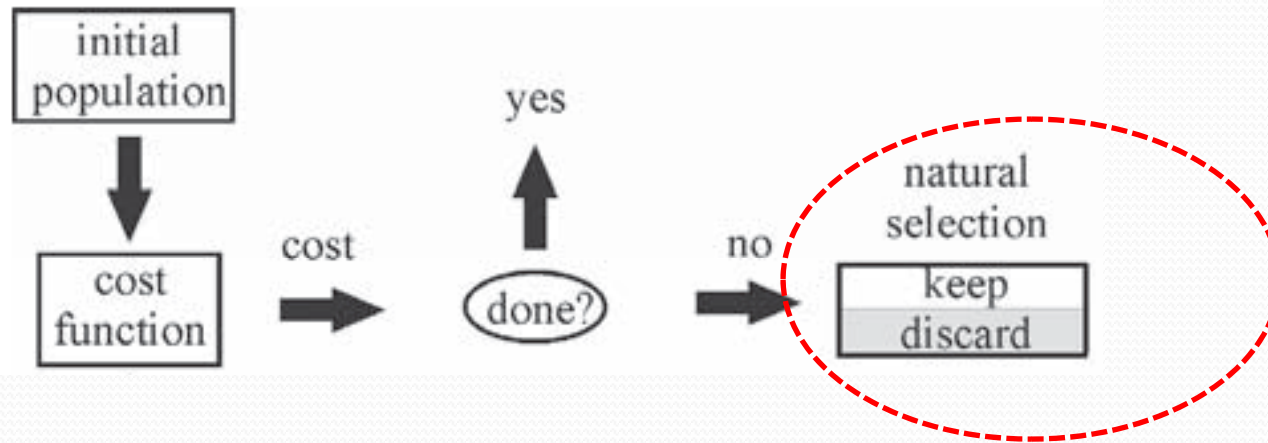


X

cost

1	1	0	0	1	1	0	0	$204/256 = 0.79688$	0.13863
0	0	0	1	1	0	0	1	$25/256 = 0.097656$	0.23756
0	0	0	0	1	1	1	1	$15/256 = 0.058594$	0.15554
1	1	1	0	1	0	0	1	$233/256 = 0.91016$	0.064732
0	0	1	0	1	0	0	1	$41/256 = 0.16016$	0.33552
1	0	1	0	0	0	1	1	$163/256 = 0.63672$	0.26391
1	1	0	1	0	1	0	1	$213/256 = 0.83203$	0.11392
1	0	0	0	1	0	1	0	$138/256 = 0.53906$	0.33987
0	0	1	0	1	1	0	1	$45/256 = 0.17578$	0.35407
1	1	0	0	0	0	1	1	$195/256 = 0.76172$	0.16467

natural selection



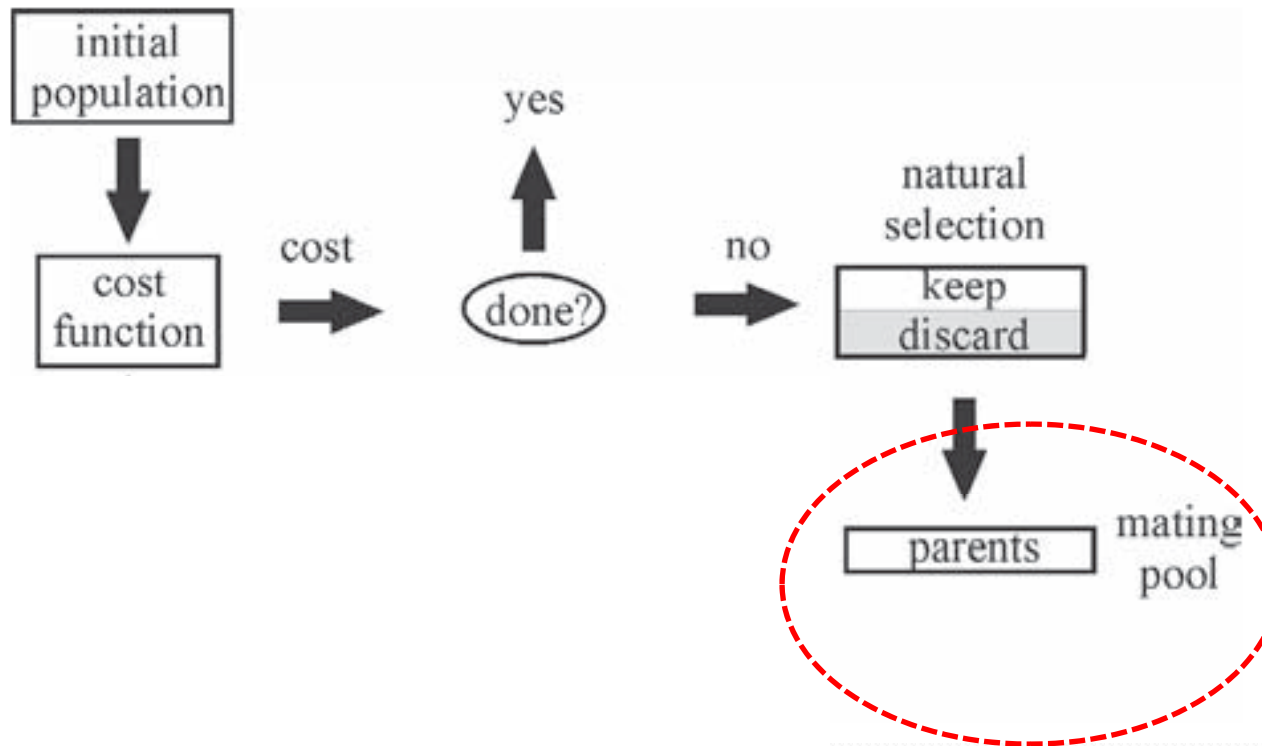
Choose 4 best:

X								cost		
1	1	0	0	1	1	0	0	$204/256 = 0.79688$	0.13863	
0	0	0	1	1	0	0	1	$25/256 = 0.097656$	0.23756	
0	0	0	0	1	1	1	1	$15/256 = 0.058594$	0.15554	
1	1	1	0	1	0	0	1	$233/256 = 0.91016$	0.064732	
0	0	1	0	1	0	0	1	$41/256 = 0.16016$	0.33552	←
1	0	1	0	0	0	1	1	$163/256 = 0.63672$	0.26391	←
1	1	0	1	0	1	0	1	$213/256 = 0.83203$	0.11392	
1	0	0	0	1	0	1	0	$138/256 = 0.53906$	0.33987	←
0	0	1	0	1	1	0	1	$45/256 = 0.17578$	0.35407	←
1	1	0	0	0	0	1	1	$195/256 = 0.76172$	0.16467	

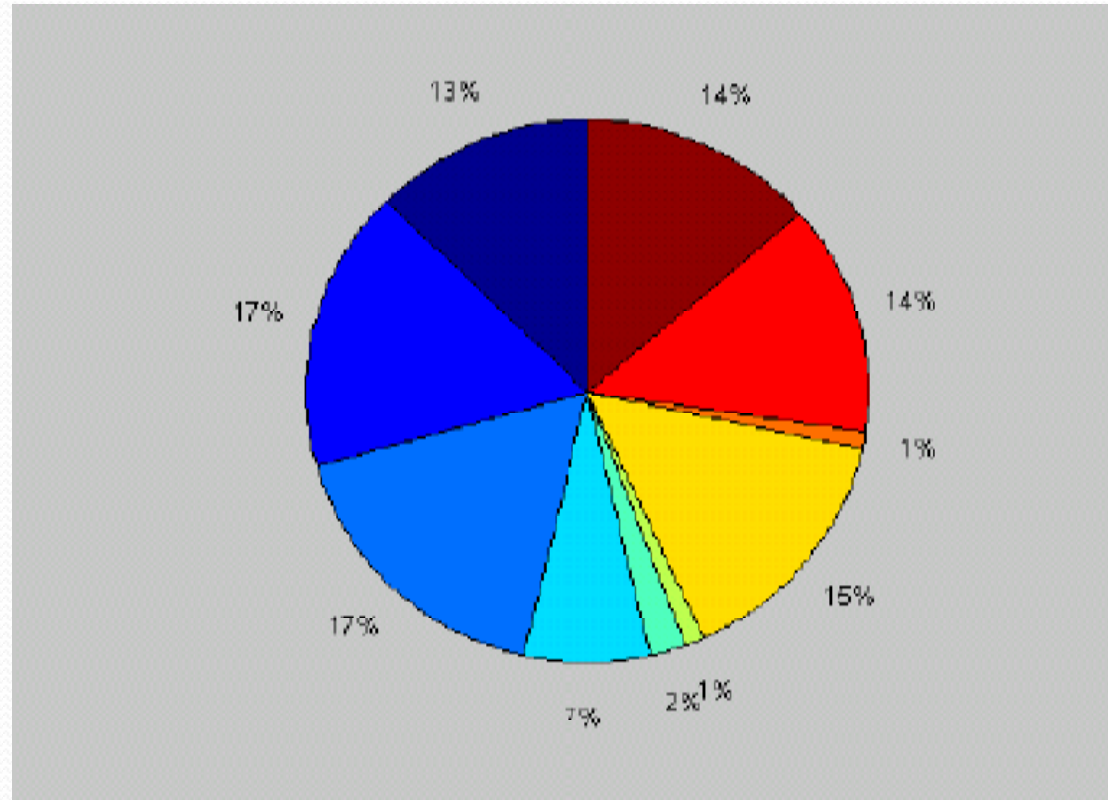
[cost ind]=sort(cost)

0.064732	4
0.11392	7
0.13863	1
0.15554	3
0.16467	10
0.23756	2
0.26391	6
0.33552	5
0.33987	8
0.35407	9

Mate selection



Roulette Wheel Selection → for pairing (father and mother)



Crossover

For binary chromosomes, uniform crossover is the most general procedure.

A mask that consists of ones and zeros is generated for each set of parents.

The mask has the same number of bits as the parent chromosomes.

Uniform crossover:

$$\text{mask} = \text{round}(\text{rand}(1, \text{nvar} * \text{nbit}))$$

Two children:

$$\text{offspring1} = \text{mask} * \text{mother} + \text{not}(\text{mask}) * \text{father}$$
$$\text{offspring2} = \text{not}(\text{mask}) * \text{mother} + \text{mask} * \text{father}$$

To demonstrate the concept of binary crossover, consider the following two parents:

mother = [1 0 1 0 1 0 1 0 1 0 1 0]
father = [1 1 1 1 1 1 0 0 0 0 0 0]

If the mask for single point crossover is given by

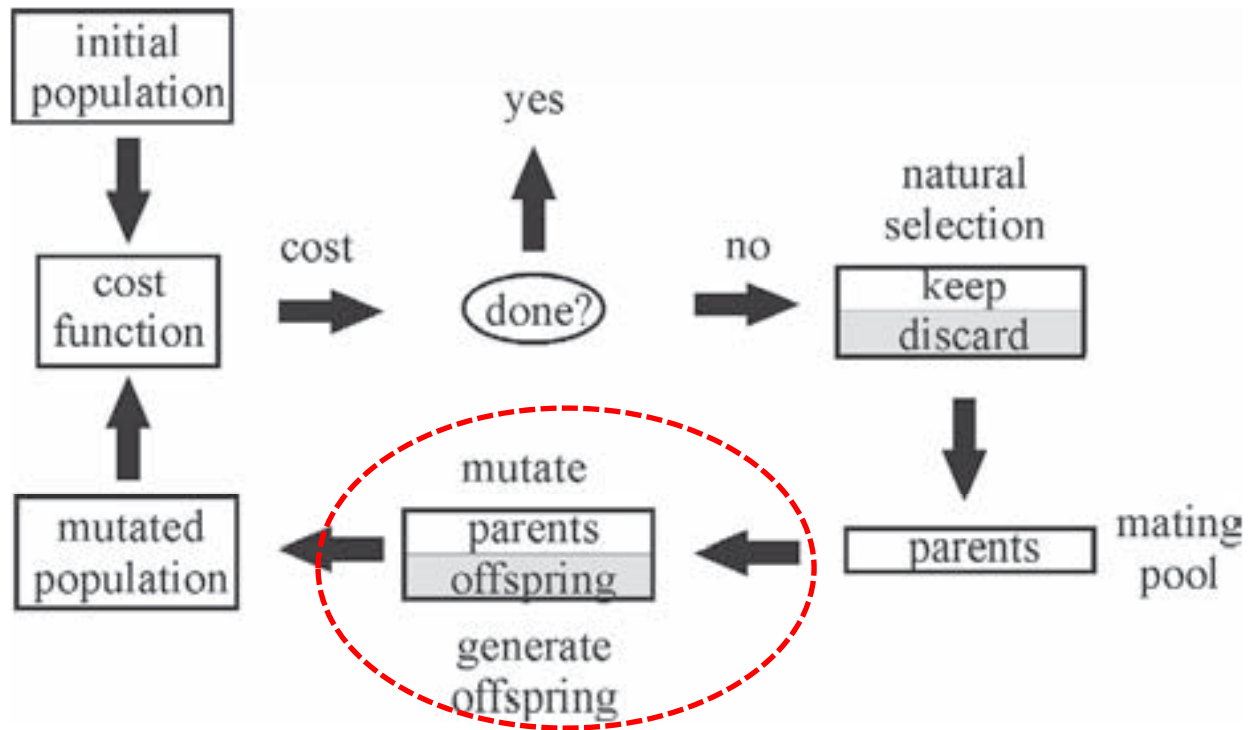
mask = [1 1 1 0 0 0 0 0 0 0 0 0]

then the offspring are

offspring1 = [1 0 1 1 1 1 0 0 0 0 0 0]

offspring2 = [1 1 1 0 1 0 1 0 1 0 1 0]

Mutation

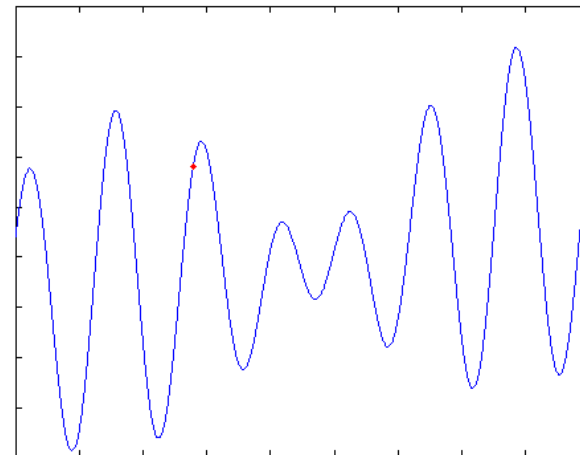
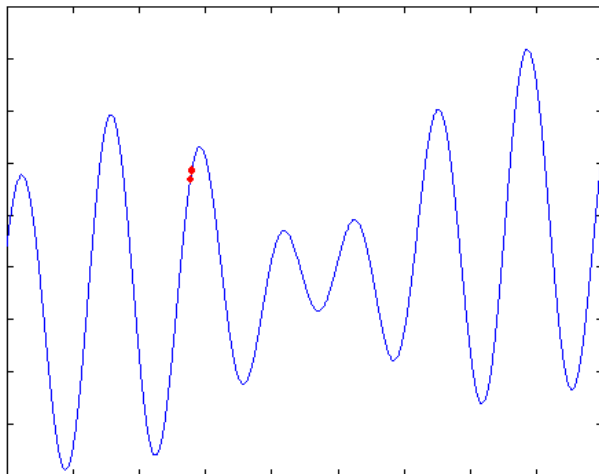
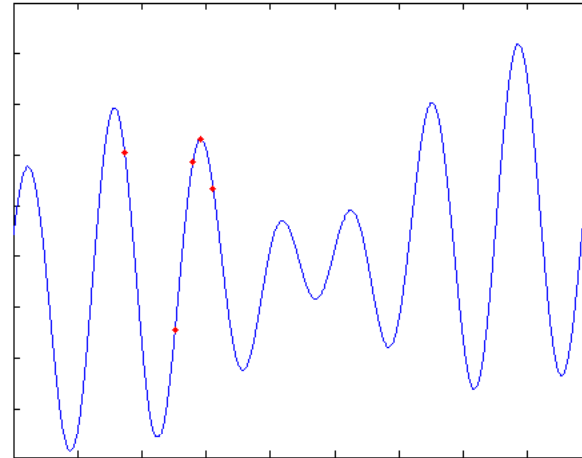
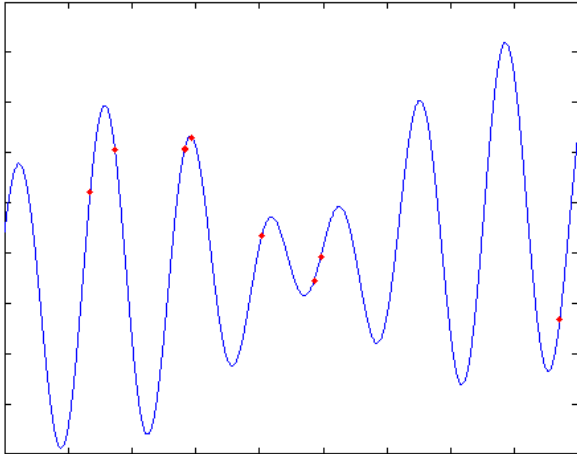


Mutation: Local Modification

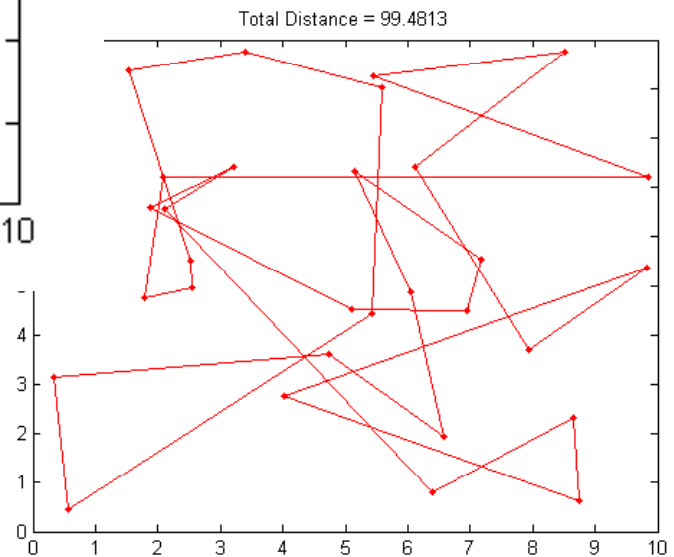
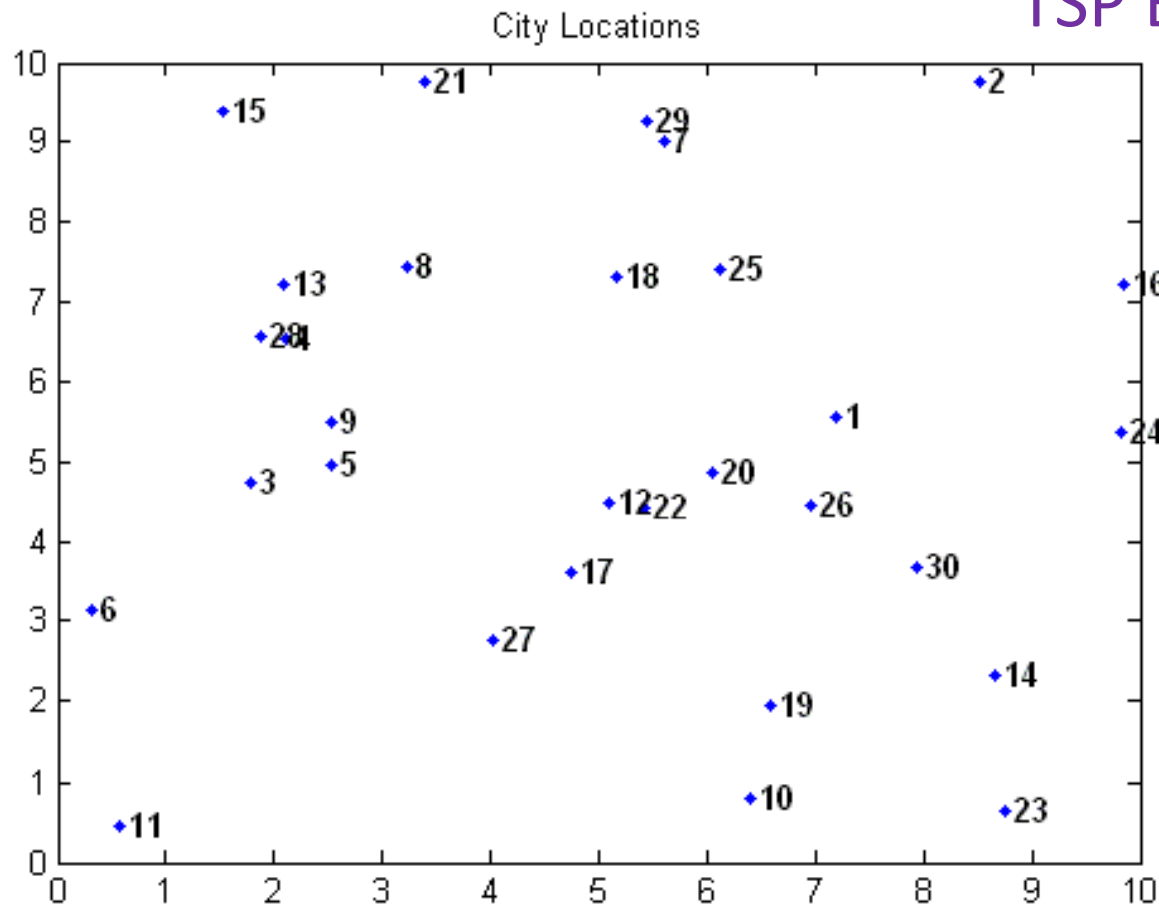
Before: (1 0 1 1 0 1 1 0)
After: (0 1 1 0 0 1 1 0)

- Causes movement in the search space (local or global)
- Restores lost information to the population

$$f(x)=x+10*\sin(5*x)+7*\cos(4*x)+\sin(x);$$

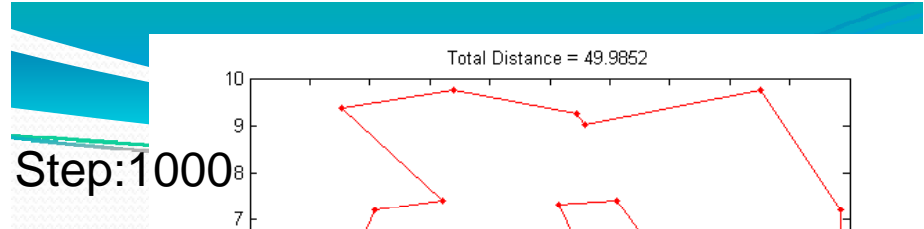
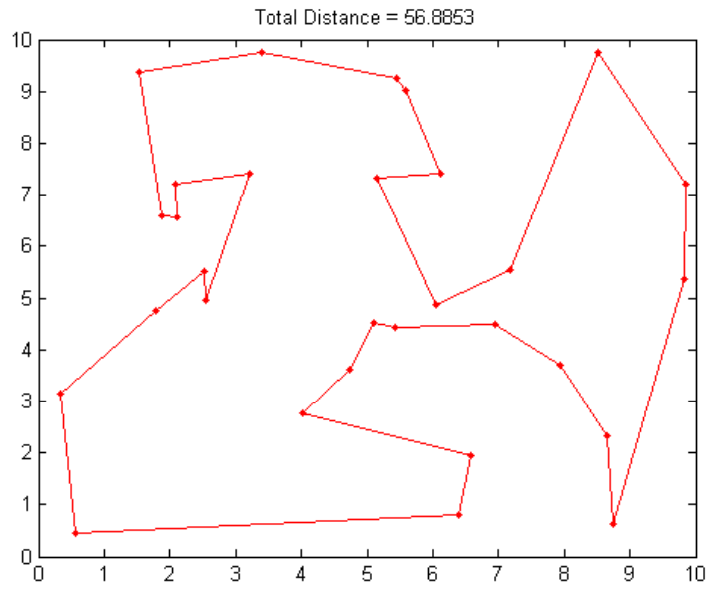


TSP Example: 30 Cities

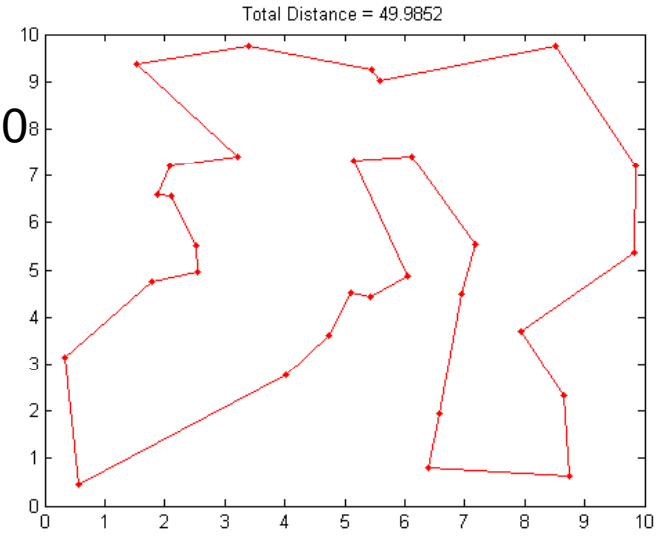




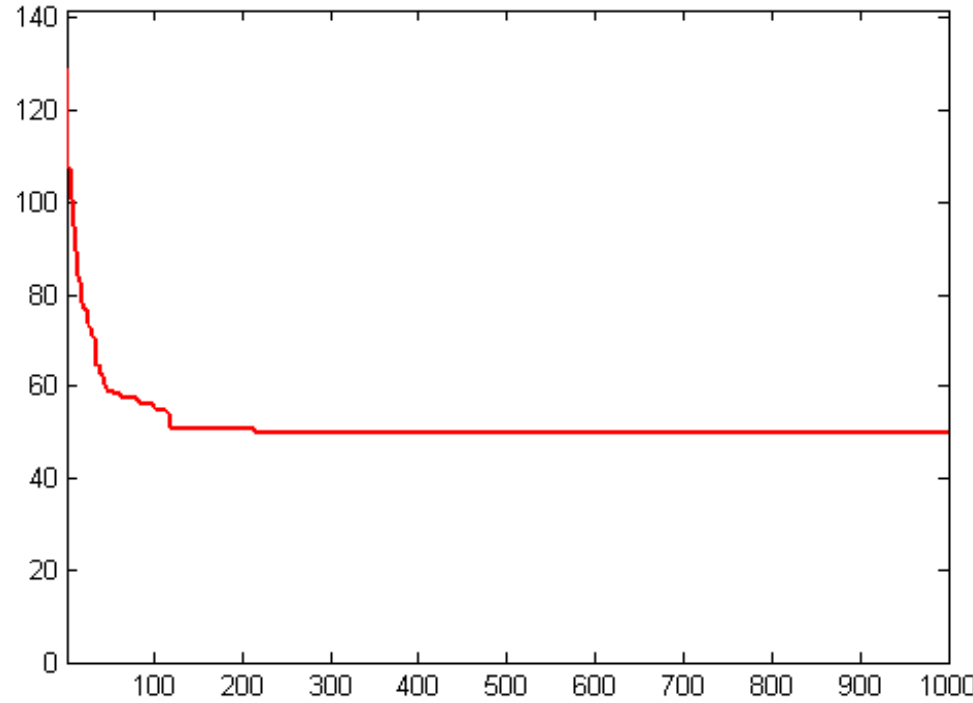
Step:100



Step:1000



Best Solution History



Some GA Application Types

Domain	Application Types
Control	gas pipeline, pole balancing, missile evasion, pursuit
Design	semiconductor layout, aircraft design, keyboard configuration, communication networks
Scheduling	manufacturing, facility scheduling, resource allocation
Robotics	trajectory planning
Machine Learning	designing neural networks, improving classification algorithms, classifier systems
Signal Processing	filter design
Game Playing	poker, checkers, prisoner's dilemma
Combinatorial Optimization	set covering, travelling salesman, routing, bin packing, graph colouring and partitioning